Handset API Overview for Vodafone 360 Phones

1. INTRODUCTION 2
1.1 ABOUT THIS DOCUMENT 2
1.2 WHO SHOULD READ THIS DOCUMENT 2
2. INTRODUCTION TO JIL ON VODAFONE PHONES 2
2.1 SPECIFICATIONS, VERSIONS, AND COMPATIBILITY 3
3. JIL HANDSET APIS 3
3.1 API SUMMARY 3
3.2 FIND AND LAUNCH NATIVE APPLICATIONS 4
3.2.1 `Widget.Device.getAvailableApplications()` 5
3.2.2 `Widget.Device.launchApplication()` 5
3.2.3 Well known applications 6
3.3 GET GPS POSITION INFORMATION 6
3.3.1 `Widget.Device.DeviceStateInfo.requestPositionInfo()` 6
3.3.2 `Widget.Device.DeviceStateInfo.onPositionRetrieved()` 7
3.3.3 `Widget.Device.PositionInfo` 8
3.3.4 GPS Position properties 8
4. RESOURCES 9
5. REVISION HISTORY 9

1. Introduction

The first JIL APIs for mobile widgets are now available on new phones from Vodafone, which are introduced as part of the new Vodafone 360 services. The first JIL APIs enable widgets to interact with native applications that are present on phones, and to access GPS positioning information on the phone.

For more about Vodafone 360 and specific phones from Vodafone that support JIL widgets, see the **Resources** section at the end of this document.

1.1 About this document

This document is the starting point for mobile developers using the JIL SDK to target widget enabled phones on Vodafone.

It describes the JIL API subset that is now available, provides code snips that show how to use it, and provides more context about the status and availability of JIL.

1.2 Who should read this document

This document is targeted at developers who want to extend their existing mobile widgets to take advantage of the new JIL APIs.

It also targets developers keen to mobilise their existing desktop widgets, or planning to implement new mobile widgets.

2. Introduction to JIL on Vodafone phones

The first JIL APIs are now available natively on the new Vodafone 360 phones.

JIL extends the functionality available to mobile widgets on supported phones. The full JIL API specification defines a comprehensive set of phone, network, and service APIs. The APIs are intended to work consistently on all phones that support the specification, regardless of network operator, phone manufacturer, browser engine, or phone operating system.

JIL generally aligns with W3C widget specifications. Unlike the W3C specifications which are still evolving, the JIL specification is stable, released, and is now being rolled out by supporting network partners, meaning that widget developers have a well-defined platform to target, a clear market, and a guarantee of future compatibility.

The first JIL APIs from Vodafone define a small but stable baseline subset of the complete JIL APIs.

2.1 Specifications, versions, and compatibility

JIL specifications define two aspects of widget behaviour:

- Widget packaging and configuration, defining the widget structure including standard file names, types, formats, and content required to enable a widget to install and run on a target phone
- Runtime APIs, defining the extension objects, methods, and properties that are implemented in the runtime engine, and how widgets can access them

The latest JIL SDK version 1.1 supports the JIL 1.1.4 specification.

The first Vodafone phones to support JIL natively are based on the JIL 1.1.1 specification.

In practice this difference will have no impact on developers:

- The packaging specifications are identical between versions, and follow the W3C (draft) specifications
- The small number of API changes between the two specifications fall outside the subset of APIs implemented in the first Vodafone phones

The more significant factor for developers is that compared to the full JIL API set, the initial subset of Vodafone supported APIs is small. This baseline is expected to evolve following the JIL 1.1.4 specification, giving developers a clear forward roadmap.

3. JIL Handset APIs

JIL extends the JavaScript widget namespace with new objects, methods, and properties.

> ***Note**: The JIL* `Widget` *object (capitalised "W") is the top level container for all JIL extensions, and defines a distinct namespace from the W3C defined* `widget` *(uncapitalised "w") namespace. The two are compatible (i.e. they do not interfere with each other), and are non-overlapping (i.e. they contain different objects), but should not be confused.*

All methods, objects, and properties are therefore defined and available as part of the runtime widget platform on supporting phones.

- API functions are invoked as methods of the object that defines them, using conventional 'dot' operator syntax and a fully qualified name
- API properties are accessed as properties of their encapsulating object, which is returned by the appropriate method invocation

3.1 API summary

The first available JIL APIs from Vodafone support two areas of functionality:

- Interaction with local native applications, enabling widgets to find and launch native applications on a phone
- GPS position information, enabling widgets to access data from an onboard GPS receiver

The table below briefly summarises the available objects, methods, and properties.

> ***Note**: The API implementation does not consistently follow the common JavaScript naming convention that capitalised names represent object constructors; for example,*

the `Widget`, `Device`, *and* `DeviceStateInfo` *objects below are not necessarily constructors.*

JavaScript methods, objects, and properties	Functionality
`Widget.Device.getAvailableApplications()`	Get the list of native applications that may be launched programmatically Returns an array of strings, the local application names – see **Well Known Applications**, below
`Widget.Device.launchApplication()`	Launch an application by name with a supplied parameter string
`Widget.Device.DeviceStateInfo.requestPositionInfo()`	Request GPS position properties Triggers the asynchronous method that returns position information, returns no value. GPS info is returned to the `onPositionRetrieved()` callback method, which the caller should implement
`Widget.Device.DeviceStateInfo.onPositionRetrieved()`	Callback method invoked in response to `requestPositionInfo()`, writes current position properties into a `PositionInfo` object that is passed as its first parameter
`Widget.Device.PositionInfo`	Object that encapsulates GPS position properties
`Widget.Device.PositionInfo.latitude`	GPS position property
`Widget.Device.PositionInfo.longitude`	GPS position property
`Widget.Device.PositionInfo.altitude`	GPS position property
`Widget.Device.PositionInfo.accuracy`	GPS position property
`Widget.Device.PositionInfo.altitudeAccuracy`	GPS position property
`Widget.Device.PositionInfo.timestamp`	GPS position property

Table 1 - JavaScript methods, objects, and properties of the JIL Handset API on Vodafone

3.2 Find and launch native applications

The ability to find and launch native applications is a powerful feature, enabling JIL widgets to interact with system services like the File Manager and Phone Settings applications, to access the network using Messaging and Mail applications, and to interact with user content through applications like the Picture Gallery.

3.2.1 `Widget.Device.getAvailableApplications()`

Purpose:

Get the names of native applications that are available on the phone.

Value returned	Method name	Parameters
Array of strings	`getAvailableApplications()`	None

Details:

Returns an array of strings, the application names on the local device.

See **Well known applications**, below, for some common name strings.

`Device` is a sub-object of the top-level `Widget` object.

Error conditions:

N/A

Code example:

The following code assumes a widget that has an HTML page element with the property `id="textdivshowResult"`:

```
// Show the list of available applications in an HTML element
var v = Widget.Device.getAvailableApplications();
v.join(", ");
document.getElementById("textdivshowResult").innerHTML = v;
```

3.2.2 `Widget.Device.launchApplication()`

Purpose:

Launch a named native application with start parameters.

Value returned	Method name	Parameters
`Void`	`launchApplication(<String> application, <String> startParameter)`	`<String> application` – required, the application name, a simple string `<String> startParameter` – required, parameters to be supplied to the application

Details:

The application string is the name of the application as returned by `getAvailableApplications()`.

The `startParameter` string is passed to the native application.

> ***Note**: The second parameter **must always** be provided; if there are no desired parameters then supply the empty string "".*

Returns `void`.

`Device` is a sub-object of the top-level `Widget` object.

See **Well known applications**, below, for some common name strings and for the list of parameters recognised by specific applications.

Error conditions:

An error code will be thrown if the second parameter is not supplied by the caller.

Code example:

The following code will launch the JIL emulator Calendar application.

The launched application automatically receives focus and runs as the foreground application, moving the widget that launched it into the background. The widget can only recover foreground when the launched application is closed by the end-user.

```
// In the JIL emulator the calendar app is called "Calendar"
// The second parameter is always required
Widget.Device.launchApplication("Calendar", "");
```

3.2.3 Well known applications

The table below shows some of the common application name strings that are returned on the JIL emulator and on the first Vodafone 360 phones.

Application	Name string returned on Emulator	Name string returned on phone
Calendar	Calendar	com.samsung.calendar
Camera	Camera	com.samsung.camera
Clock Alarms	Alarm	com.samsung.worldclock
Email Application	Mail	com.samsung.email
File Manager	File Manager	com.samsung.myfiles
Messaging Application	Messaging	com.samsung.message
Picture Gallery	Pictures	com.samsung.gallery
Phone Settings	Settings	com.samsung.setting
Web Browser	Browser	com.samsung.internet

Table 2 – Well known application names on Vodafone 360 phones

3.3 Get GPS position information

Accessing GPS position information enables JIL widgets to become fully location aware, creating opportunities to integrate location context with local services on the phone and with network based content.

> ***Note**: All GPS data is supplied by the onboard GPS receiver. The usual comments about using GPS therefore apply – you need to consider potential latency issues in getting satellite fixes, as well as GPS coverage issues, for example if the phone is being used inside a building.*

3.3.1 `Widget.Device.DeviceStateInfo.requestPositionInfo()`

Purpose:

Request GPS position information from the onboard receiver.

Value returned	Method name	Parameters
`Void`	`requestPositionInfo(<String> method)`	`<String> method` – required, the desired location method Currently only GPS positioning

		is supported, the value should therefore always be `gps`

Details:

The `method` parameter specifies the location technology to be used – currently only GPS positioning is supported, the value should therefore always be `gps`.

Asynchronous method, returns immediately with no value – GPS position information is returned in a `PositionInfo` object to the `onPositionRetrieved()` callback method the caller must implement.

`Device` is a sub-object of the top-level `Widget` object.

`DeviceStateInfo` is a sub-object of the `Device` object.

See `Widget.Device.DeviceStateInfo.onPositionRetrieved()`.

Error conditions:

N/A

Code example:

See `onPositionRetrieved()`, below.

3.3.2 `Widget.Device.DeviceStateInfo.onPositionRetrieved()`

Purpose:

Callback method used to return requested position information.

Value returned	Method name	Parameters
`void`	`onPositionRetrieved(<Positioninfo> locationinfo, <String> method)`	`<Positioninfo> locationinfo` – required, the `positionInfo` object through which the current GPS data are available `<String> method` – the location method used to get the position information Currently only GPS positioning is supported, the value should therefore always be `gps`

Details:

The `PositionInfo` parameter is a `PositionInfo` object into which position properties will be written, describing the phone's current geographical location.

The `method` parameter specifies the location method that was used to get the position information – currently only GPS positioning is supported, therefore in practice this parameter is not required.

> ***Note**: The JIL specification defines possible position information technologies as* `cellid`*,* `gps` *and* `agps` *(assisted GPS).*

`Device` is a sub-object of the top-level `Widget` object.

`DeviceStateInfo` is a sub-object of the `Device` object.

`PositionInfo` is a sub-object of the `DeviceStateInfo` object.

See `Widget.Device.DeviceStateInfo.requestPositionInfo()`.

Error conditions:

N/A

Code example:

The following code demonstrates the steps required to create the necessary objects for requesting location information. This includes defining the callback method to which the data will be returned in a `PositionInfo` object.

The code assumes a widget that has an HTML page element with the property `id="textdivshowResult"`, into which we write the retrieved latitude and longitude properties.

The last line is the method invocation, which triggers the callback; the `gps` parameter should always be supplied to `requestPositionInfo()`:

```
// Instance the method owning objects
var wdd = window.Widget.Device.DeviceStateInfo;
// Define a callback
wdd.onPositionRetrieved = function (posInfo) {
    document.getElementById("textdivshowResult").innerHTML =
      "latitude: " + posInfo.latitude + " longitude: " +
          posInfo.longitude;
};
// Make the request
wdd.requestPositionInfo("gps");
```

3.3.3 `Widget.Device.PositionInfo`

Purpose:

Object containing position properties.

Namespace	Object name	Properties
`Device`	`PositionInfo`	GPS position properties, see table below

Details:

An instance of this object is declared and passed in the `onPositionRetrieved()` callback method used to receive position data using `requestPositionInfo()`.

Currently only GPS data types are supported.

The `PositionInfo` parameter is a `PositionInfo` object into which position properties will be written, describing the phone's current geographical location.

`Device` is a sub-object of the top-level `Widget` object.

`PositionInfo` is a sub-object of the `Device` object.

See also `Widget.Device.DeviceStateInfo.requestPositionInfo()` and `Widget.Device.DeviceStateInfo.onPositionRetrieved()`.

3.3.4 GPS Position properties

The `PositionInfo` object contains the following GPS properties that define the current phone location:

Type of value	Property name	Description
Number	`latitude`	Latitude in degrees using the World Geodetic System 1984 (WGS84) datum
Number	`longitude`	Longitude in degrees using the World Geodetic System 1984 (WGS84) datum
Number	`altitude`	Altitude in metres using the World Geodetic System 1984 (WGS84) datum
Number	`accuracy`	The horizontal accuracy of the position in metres, or `null` if not available
Number	`altitudeAccuracy`	The vertical accuracy of the position in metres, or `null` if not available
Date	`timeStamp`	The date and time when the location was established

4. Resources

Vodafone 360 is a new service for your phone, PC and Mac that brings together your friends, communities, apps, games and music in one place. For more information, visit 360.com.

The new Vodafone 360 phones are the first phones from Vodafone to support JIL widget APIs. The models are:

- Vodafone 360 H1 by Samsung (480x800 display)
- Vodafone 360 M1 by Samsung (240x400 display)

JIL is the **Joint Innovation Lab**, a joint venture between leading global network operators – Vodafone, China Mobile, SoftBank Mobile, and Verizon Wireless.

JIL enables access to a consistent set of phone, network, and service APIs that are intended to work consistently on all phones that support the specification, regardless of network operator, phone manufacturer, browser engine, or phone operating system.

For more information and to keep up to date with the latest about JIL on Vodafone, visit jil.org – register for free to download the JIL SDK and get started with JIL mobile widgets. The JIL SDK, currently at version 1.1, is Eclipse-based and is supported on Windows and Linux platforms.

5. Revision History

Version	Date	Comments
1.0	21/09/2009	First publication